Objective

The objective of this workshop is to

a. Build a container image
b. Deploy the image to a Kubernetes cluster
c. Make the deployment accessible from the outside of the cluster by exposing it as a load balancer

Setup

a. Create a DockerHub account at https://hub.docker.com if you do not have one
b. Create a directory call `workshop02` in your Git repository.

Workshop

In this workshop you will be building a container image from a given application. After building the container, you will deploy it to a Kubernetes cluster.

Task 1

In this task, you will be building a container image to serve an Angular application. The following are the prescribed steps to build and run the application.

i. Build Angular

You are given an Angular application to deploy. To deploy this application, you will need to perform the following

a. Build the Angular application
b. Serve the Angular application with a web server (Caddy)

The following are the steps to build the Angular application, assuming that you have Node installed

a. Install Angular CLI

```
npm install -g @angular/cli
```

b. Install the application's dependencies, to be perform in the application's directory

```
npm ci
```

c. Build the application.

```
ng build
```

The build artifacts will be in `dist/angular/browser` directory after a successful build.

ii. Serving the Application with Caddy

We will be using Caddy (https://caddyserver.com/) to serve the application. In the caddy directory, you will find the following:

- `Caddyfile` – configuration for running the Caddy. The configuration file sets the following:
  - Set Caddy to listen on port 3000 (default). The port can be change by setting the environment variable `CADDY_PORT` eg. `CADDY_PORT=5000` sets the port to 5000
  - Serve documents from `html` (document root)
  - `error.html` will be served when a requested resource is not found (404).

Create a directory called `html`.

Copy the contents of `dist/angular/browser/` and `error.html` into `html` directory.

Start Caddy with the following command from within `caddy` directory:

```
caddy run --config Caddyfile
```

You can now access the application by typing http://localhost:8080 into your browser's address bar.

iii. Build and push the image

Use the above steps to build a container image and push it to Docker Hub (or any other container registry).

On platforms that are not x86 eg. OSX or Windows using ARM64 processors, you will need to build a x86 image because the target platform, Kubernetes nodes, are x86.

Use the following command to build for x86 (in a single line)

```
docker buildx build \
    --platform linux/amd64 \
    -t <your image name/tag> .
```

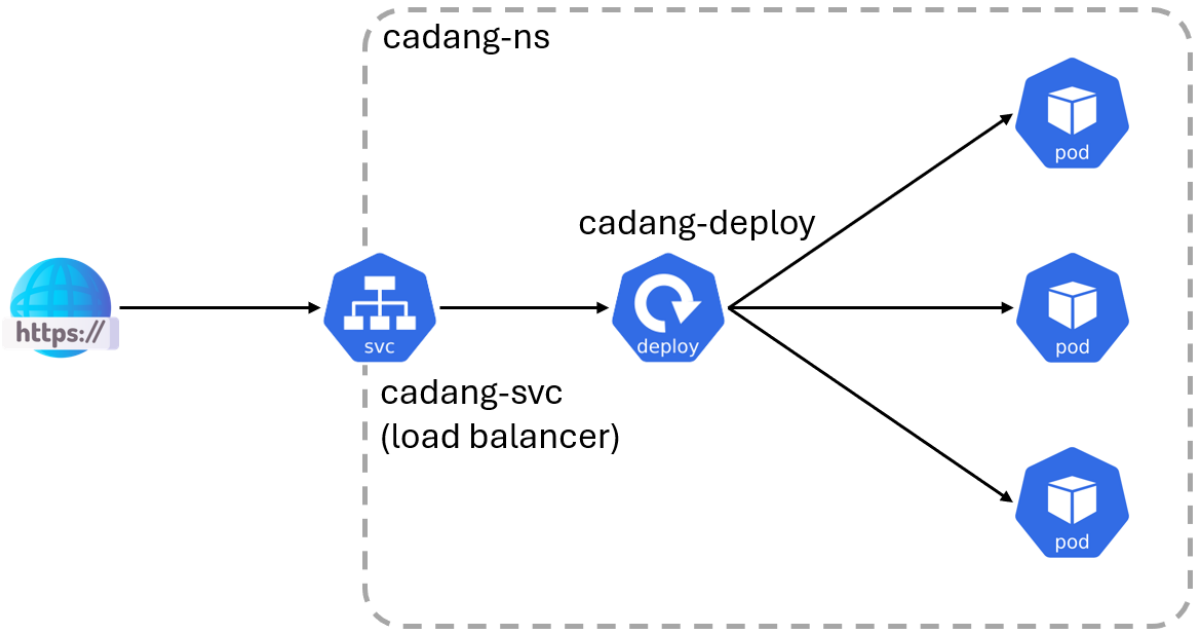
Task 2

Use the image from Task 1 to deploy into your Kubernetes cluster according to the following requirements

a. The application must be deployed into its own namespace called `cadang-ns`.
b. Create a 3-replica deployment called `cadang-deploy`.
c. Expose the deployment with a load balancer service called `cadang-svc`. Expose the service port on port 80.

Access you application on with the load balancer's IP address.

The following diagram shows the architecture of Task 2.



Task 3

i. Upgrade the application

Update the application with the following image.

```
chukmunnlee/cfdsa-fortune:v1
```

Ensure that there are always 2 instances of the application available during the upgrade.

Note: The `cfdsa-fortune` application listens on port 3000.

Test the upgrade after rolling out `cfdsa-fortune:v1`.

ii. Rollback the application.

Rollback `cadang-deploy` to your image. Test the application after the rollback.

Submission

Create a directory called `workshop02` inside your repo. Placed all the file for this workshop inside `workshop02` directory.

Submit the following files in `workshop02` directory:

a. Kubernetes deployment and service files
b. Any other files